Reducibility

Reading

Section 5.1

Reducibility in general

So far we have established essentially problems that are Turing-recognizable not decidable: A_{TM} and NONSELFACC. In this section we will build many more, using the idea of reducibility:

> A **reduction** is in general a way of converting one problem to another problem in such a way that a solution to the second problem can be used to solve the first problem.
>
> In our context, problems are represented by languages, and we will say for languages A and B that A *reduces to* B if we can use a solution of B to provide a solution for A.
>
> Essentially reducibility says that if A reduces to B, then solving A cannot possibly be harder than solving B.

We already used this idea: NONSELFACC reduces to A_{TM}: If we have a decider (i.e. a "solution") to A_{TM}, then we could use it to get a decider (a "solution") for NONSELFACC. Since that is not possible (NONSELFACC is undecidable), we used this idea to show that A_{TM} is undecidable.

Informally we can say:

> If a language A reduces to a language B then:
>
> - If B is decidable, then A is also decidable.
> - If A is undecidable, then B is also undecidable.

We will extend this idea, to show many languages to be undecidable by reducing A_{TM} to them.

The actual Halting problem

The actual Halting Problem language consists of all pairs of a Turing Machine and an input, such that the TM halts on that input:

$$\text{HALT}_{\text{TM}} = \{\langle M, w\rangle \mid M \text{ is a TM and halts on } w\}$$

We will now show this language is undecidable. We will do so by reducing $A_TM to it:

1. Suppose that HALT_{TM} was decidable, and let us say H is a decider for it.
2. We will use H to build a decider for A_{TM} as follows.
3. On input a pair of a TM M and input w:
   - Run H on input that pair $\langle M, w\rangle$.
   - If H rejects, that means M would loop on input w, so we reject.
   - If H accepts, then it is safe to actually run M on input w.
   - If M accepts the input w, then we also accept. If it rejects then we also reject.
4. This is a decider for A_{TM}.

The Emptyness problem

We will now show the undecidability of the emptyness problem for Turing Machines:

$$E_{\text{TM}} = \{\langle M\rangle \mid M \text{ is a TM and } L(M) = \emptyset\}$$

We will prove it is undecidable in a very similar fashion, by reducing A_{TM} to it:

1. Suppose that E_{TM} is decidable, and say D is a decider for it.
2. We will use it to construct a decider for A_{TM}.
3. We start by considering a pair $\langle M, w\rangle$ of a TM and an input string w.
4. Using M and w, we now build a new TM N that does the following:
   - If it is given an input string that does not match w, it rejects right away.
   - If its input string is w, then it runs M on w and returns that result.
5. Finally, we feed this new TM N to D (rather, we feed $\langle N\rangle$ to D).
6. If D accepts, this means that there are no strings that N would accept.
   - But the only string that N could accept is w, so we are saying it doesn't accept that either.
   - So it must not be the case, that N on input w accepts.
   - But N was simulating M on input w, so it must not be the case that M accepts on input w.
   - Therefore we can reject the pair $\langle M, w\rangle$ from A_{TM} and we are done.
7. If D rejects, this means that there must have been at least one string that N would accept.
   - But the only such possible string is w, so it must be the case that N accepts w.
   - But N simulates M on w, so it must be the case that M accepts w.

- So we can accept the pair $\langle M, w\rangle$ as belonging to A_{TM} and we are done.

8. We just built a decider for A_{TM}, which is a contradiction.

A key step in the above of course is that one should be able to actually construct N from M and w. Think of how that would go.

The regularity problem

Here is another problem that we will show as undecidable, by showing that A_{TM} reduces to it. It is very instructive, in that the proof generalizes to many situations.

$$\text{REGULAR}_{\text{TM}} = \{\langle M\rangle \mid M \text{ is a TM and } L(M) \text{ is a regular language }\}$$

So this language consists of the representations of those Turing machines whose language is a regular language. We will prove this language to be undecidable.

We describe now the proof. It follows standard reducibility steps for a while.

1. We assume that $\text{REGULAR}_{\text{TM}}$ is decidable, and let us denote by R its decider.
2. We will also fix a non-regular language L, for instance $L = \{0^n1^n \mid n \geq 0\}$.
3. We need to use R to build a decider for A_{TM}. This decider operates as follows:
4. Given the input $\langle M, w\rangle$ of a TM and a string for it, we build a new TM, N, as follows:
   - If given input $x \in L$, then N accepts.
   - If given input a string $x \notin L$, then N runs M on w and returns what M would return.
5. Before we move on let us figure out the language of N.
   - If M accepts w, then the language of N will be Σ^*, all strings. This is a regular language.
   - If M does not accept w, then the language of N will be L, which is non-regular.
6. Now we run R on input the string representation of this new TM $\langle N\rangle$.
   - If R on input $\langle N\rangle$ accepts, then this means that the language recognized by N is regular. But then from step 5 this means that M better be accepting w, and we have answered the A_{TM} question for the pair $\langle M, w\rangle$ in the affirmative.
   - If R on input $\langle N\rangle$ rejects, then this means that the language recognized by N is not regular, so by step 5 it means that M should not accept w, and we have answered the question in the negative.
7. We have just demonstrated a decider for A_{TM}, which is a contradiction.

This idea extends to a tremendous degree, to what is known as Rice's theorem:

Rice's Theorem

Suppose that P is a "property" on Turing Machines (so for each TM M we have that $P(M)$ is either true or false), such that:

- P is a non-trivial property, i.e. there are some TM's that belong to it ($P(M)$ true) and some TM's that don't belong to it ($P(M)$ false).
- P only depends on the language $L(M)$ of the TM, i.e. if $L(M_1) = L(M_2)$ then $P(M_1) = P(M_2)$.

Then the language $\{\langle M\rangle \mid P(M) \text{ is true}\}$ is undecidable. We often simply define this language by P itself: We think of a property as defining a subset of all Turing Machines, by picking out those TMs for which the property is true.

In simpler terms, "Every non-trivial property of Turing Machines is undecidable".

Look at problem 5.28 and its solution for an idea.

Equivalence of TMs

Finally, we show that the problem of equivalence of TMs is undecidable, by reducing E_{TM} to it. In general, we don't always have to reduce A_{TM} to the problem we are trying to prove is undecidable. Reducing any already-known-to-be-undecidable problem to our target problem would work just as well.

$$\text{EQ}_{\text{TM}} = \{\langle M_1, M_2\rangle \mid L(M_1) = L(M_2)\}$$

This language is also undecidable. Our proof can proceed as follows. Suppose that it was not, and that D was a decider for it. We will build a decider for E_{TM} as follows:

1. First, make a TM N that simply rejects all input, so $L(N) = \emptyset$.
2. Now given a TM M, we need to decide if its language is empty.
3. All we need to do is run D on the pair $\langle M, N\rangle$.
4. We have a decider for E_{TM}!